IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPELLANTS' MAIN BRIEF ON APPEAL

APPLICANTS:	Hideki Hiura, et al.	ATTY DOCKET NO:	30014200-1027
SERIAL NO.:	09/488,909	GROUP ART UNIT:	2194
DATE FILED:	January 21, 2000	EXAMINER:	Phuong N. Hoang
INVENTION:	"METHOD FOR ENABLING MULTIPLE CONCURRENT SUBPROCESS HANDLING ON A SYSTEM USING A GLOBAL PROCESS		

Mail Stop Appeal Brief - Patents
Hon. Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

Appellants submit herewith Appellants' Main Brief on Appeal under 37 C.F.R. §41.37 in support of the Notice of Appeal mailed on February 22, 2007. The Commissioner is hereby authorized to charge the amount of $500.00 for the requisite filing fee for filing the Main Brief on Appeal to the Appellants' Attorneys' credit card. Credit Card payment for the fee is made via the electronic submission process.

The Commissioner is hereby authorized to charge any deficiency in fees associated with this communication or credit any overpayment to Deposit Account No. 19-3140. A duplicate copy of this sheet is enclosed.

Respectfully Submitted,

/Christopher P. Rauch/   (Reg. No. 45,034)
Christopher P. Rauch
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box #061080
Wacker Drive Station - Sears Tower
Chicago, IL 60606-1080
Telephone 312/876-2606
Customer #58328
Attorneys for Appellants

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPELLANTS' MAIN BRIEF ON APPEAL

APPLICANTS:	Hideki Hiura, et al.	ATTY DOCKET NO:	30014200-1027
SERIAL NO.:	09/488,909	GROUP ART UNIT:	2194
DATE FILED:	January 21, 2000	EXAMINER: Phuong N. Hoang	
INVENTION:	"METHOD FOR ENABLING MULTIPLE CONCURRENT SUBPROCESS HANDLING ON A SYSTEM USING A GLOBAL PROCESS		

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

In accordance with the provisions of 37 C.F.R. §41.37, Appellants submit this Main Brief on Appeal pursuant to the Notice of Appeal mailed on February 22, 2007 in the above-identified application.

I.   REAL PARTY IN INTEREST:

The real party in interest in the present appeal is the Assignee, Sun Microsystems, Inc.. The assignment was recorded in the U.S. Patent and Trademark Office at Reel 010810, Frame 0001.

II.   RELATED APPEALS AND INTERFERENCES:

Appellants are not aware of any related appeals or interferences.

III.   STATUS OF CLAIMS:

Claims 1-21 are pending in the application.

Claims 1-11 and 15-21 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Hetherington (U.S. Patent No. 6,275,810)* in view of *Kaufman (U.S. Patent No. 5,313,647)*.

Claims 12-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Kaufman (U.S. Patent No. 5,313,647)* in view of *Hetherington (U.S. Patent No. 6,275,810)*.

The present appeal is directed to claims 1-21, which were finally rejected in an Office Action dated September 22, 2006.

A copy of claims 1-21 is appended hereto as the Claims Appendix.

IV.   STATUS OF AMENDMENTS:

An Amendment After Final was filed on December 12, 2006 in response to the final Office Action dated September 22, 2006. In an Advisory Action dated February 8, 2007, the Examiner identified that the Amendment After Final has been entered for purposes of appeal.

Thus, all amendments have been entered in this application.

V.   SUMMARY OF CLAIMED SUBJECT MATTER:

Claims 1-21 are currently pending. Claims 1, 8, 12, and 15 are the only pending independent claims under consideration. Claims 2-7, 9-11, 13-14, and 16-21 depend directly or indirectly from claims 1, 8, 12, or 15.

Independent claims 1, 8, 12, and 15 are summarized below.

Claim 1:

Claim 1 relates to a method, in a computer system, for providing for concurrent subprocessing of a master process. By way of example and not limitation, the means for performing this method may be a virtual memory separator (identified by numeral 232 of Figure 2) that is part of an operating system. The virtual memory separator 232 interfaces with a master process when a user-specific operation is encountered. (*See* master process 302 and user-specific operation 310 of Figure 3; steps 410, 412, and 414 of Figure 4; and pages 5, line 20 to page 7, line 22 of the Specification). The virtual memory separator 232 facilitates mapping a user-specific process so that it overlays the virtual addresses of the master process (*e.g.*, user specific processes 304, 306, and 308 are mapped to master process 302). (*See* Figure 3 and page 5, line 20 to page 6, line 18 of the Specification). The virtual memory separator 232 further processes the user-specific operation in the user-specific process. (*See* steps 410, 412, and 414 of Figure 4 and page 6, line 9 to page 7, line 22 of the Specification).

<u>Claim 8:</u>

A computer-readable medium comprising computer instructions that facilitate concurrent handling of subprocesses in a system that utilizes a global process, wherein the instructions, when executed, cause the system to perform the steps of:

interfacing with the global process when a user-specific operation is encountered;

mapping a plurality of concurrent user-specific processes, wherein each user-specific process is mapped to virtual addresses that are equivalent to virtual addresses of the global process; and

processing the user-specific operation in one of the user-specific processes.

Claim 8 relates to a computer-readable medium comprising computer instructions that facilitate concurrent handling of subprocesses in a system that utilizes a global process. (*See* page 10, lines 11-17). The instructions, when executed, cause the system to perform the steps described below.

By way of example and not limitation, the means for performing this method may be a virtual memory separator (identified by numeral 232 of Figure 2) that is part of an operating system. The virtual memory separator 232 interfaces with a gobal process when a user-specific operation is encountered. (*See* master process 302 and user-specific operation 310 of Figure 3; steps 410, 412, and 414 of Figure 4; and pages 5, line 20 to page 7, line 22 of the Specification). The virtual memory separator 232 facilitates mapping a plurality of concurrent user-specific processes, wherein each user-specific process is mapped to virtual addresses that are equivalent to virtual addresses of the global process (*e.g.*, user specific processes 304, 306, and 308 are mapped to global process 302). (*See* Figure 3 and page 5, line 20 to page 6, line 18 of the Specification). The virtual memory separator 232 further processes the user-specific operation in the user-specific process. (*See* steps 410, 412, and 414 of Figure 4 and page 6, line 9 to page 7, line 22 of the Specification).

<u>Claim 12:</u>

Claim 12 is directed to a computer system for enabling concurrent multiple subprocess handling in a global process environment. The computer system includes a global process (*e.g.* master process 302), and a virtual memory separator (*e.g.* virtual memory separator 232) that maps a user-specific process (*e.g.* locale-dependent process 304) to virtual addresses that mirror virtual addresses of the global process. (*See* page 5, line 20 to page 7, line 22 of the Specification and Figures 3-4). The user-specific process having an interface that mirrors an interface of the global process. (*See* page 6, lines 9-18 of the Specification).

<u>Claim 15:</u>

Claim 15 relates to An apparatus for conducting multi-user concurrent handling of subprocesses. By way of example and not limitation, the means for performing this method may be a virtual memory separator (identified by numeral 232 of Figure 2) that is part of an operating system. The virtual memory separator 232 interfaces with a master process when a user-specific operation is encountered. (*See* master process 302 and user-specific operation 310 of Figure 3; steps 410, 412, and 414 of Figure 4; and pages 5, line 20 to page 7, line 22 of the Specification). The virtual memory separator 232 facilitates mapping a user-specific process so that it overlays the virtual addresses of the master process (*e.g.*, user specific processes 304, 306, and 308 are mapped to master process 302). (*See* Figure 3 and page 5, line 20 to page 6, line 18 of the Specification). The virtual memory separator 232 further processes the user-specific operation in the user-specific process. (*See* steps 410, 412, and 414 of Figure 4 and page 6, line 9 to page 7, line 22 of the Specification).

It is possible that the above-described elements of claim 18 may be interpreted to invoke 35 U.S.C. §112, paragraph 6. In this case, each of the elements is performed by, for example, the virtual memory separator 232 in the memory 226 and executed by processor 222.

VI.   GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL:

The ground of rejection to be reviewed on appeal are as follows:

Claims 1-11 and 15-21 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Hetherington (U.S. Patent No. 6,275,810)* in view of *Kaufman (U.S. Patent No. 5,313,647)*; and

Claims 12-14 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Kaufman (U.S. Patent No. 5,313,647)* in view of *Hetherington (U.S. Patent No. 6,275,810)*.

VII.   ARGUMENT:

As set forth below, claims 1-21 are not unpatentable under 35 U.S.C. §103(a) based on the cited references. Appellants respectfully submit the Examiner's assertions are incorrect as a matter of fact and law. Thus, for the reasons set forth below, Appellants respectfully request that this Board reverse the rejection of claims 1-21.

I.   *Heatherington* in view of *Kaufman* fails to disclose or suggest
        claims 1-11 and 15-21

Applicants respectfully submit that the Examiner has omitted one or more essential elements needed for a *prima facie* rejection. The cited art, alone or in combination, fails to teach or suggest every limitation of the claims. For example, the combination of *Hetherington* and *Kaufman* fails to teach or suggest at least mapping a user-specific process so that it overlays virtual addresses of the master/global process as recited in claims 1, 8, 12, and 15.

A.   "Forking" Is Not The Same As Overlaying Virtual Addresses

Contrary to the Examiner's assertions, "forking" fails to disclose or suggest Applicants' claimed subject matter relating to overlaying virtual addresses. The Examiner asserts that "Kaufman teaches that when spawning, the child would overlay the parent's virtual memory address (vm_folk (sic) to duplicate a parent process's virtual memory information for a child process, Col. 31 line 14-20 and Col. 2 lines 1-5)." (*See, e.g.* Advisory Action of 2/8/07, page 1). Applicants respectfully submit that, as disclosed in *Kaufman*, a child process is formed by copying but does not share or overlay the virtual addresses of the parent. Information of the parent may be copied in *Kaufman*, but the virtual address is not overlaid or the same virtual address, for example. Copying or duplication of information is not the same as sharing the same virtual addresses, and *Kaufman* equates forking with copying or duplication. For example, *Kaufman* states "[a] fork element can create a second process that initially <u>duplicates</u> a first one and can initiate generation of a new-process signal in connection with creation of the second process." (Abstract of *Kaufman*; emphasis added).

Nowhere does *Kaufman* teach or suggest that the processes share or overlay the same virtual addresses. In addition, *Kaufman* states "many computer systems, for example, those running under UNIX or UNIX-like operating systems, permit process <u>duplication, or forking</u>. Forking causes one process to <u>replicate</u>, spawning a new process. The first process, referred to as the parent, continues in the normal manner after the fork. The spawned process, or child, though initially identical to the parent, can be executed in a different manner." (Col. 2, ll. 1-5 of *Kaufman*; emphasis added). In the portion cited by the Examiner, *Kaufman* states "[t]he vm system executes the steps of the procedure vm_fork to <u>duplicate</u> a parent process's virtual memory <u>information</u> for a child process." (Col. 31, ll. 14-21 of *Kaufman*). As stated above, duplication of information is not sharing or overlaying the same virtual addresses.

Thus, *Kaufman* fails to disclose or suggest overlaying virtual addresses.

B. <u>Context Address Space Mapping Is Not Virtual Address Space Mapping</u>

Further, *Kaufman's* context address space mapping fails to disclose or suggest Applicants' claimed subject matter relating to virtual address space mapping. In response to Applicants' arguments that the duplication of information, or "forking," disclosed in *Kaufman* is not the same as overlaying the same virtual address, the Examiner further asserted that "Kaufman teaches the mapping so that the child overlaid the virtual address of the parent (vm_fork, syscall_finish_fork, vm_mapin to mapin [sic] the overlay object and mapping out a file range from a process's context address space, col. 31 lines 15-65 and col. 34 lines 10-15)" (*See* Final Official Action of 4/19/2005, p. 8, no. 24). However, the Examiner's argument with regard to mapping in and out of a process's context address space still fails to establish *prima facie* obviousness.

The portion of *Kaufman* cited by the Examiner states that "[t]he VM system executes the steps of MAPOUT as a user entry for mapping out a file range from a process's <u>context address</u> space. (*See* Col. 34, ll. 10-15 of *Kaufman*). The procedure accepts as input a handle of the map, and returns a status of the mapping procedure" (Emphasis added). However, claim 1 recites a <u>virtual</u> address overlay and not a <u>context</u> address overlay or mapping. A context address is very different from a virtual address, as explained in *Kaufman* itself:

> The memory architecture of system 10 consists of two levels of related address space: context address (CA) space and system virtual address (SVA) space. Context address space is the programmer's interface to memory. There are many context address spaces in a system. System virtual address space stores the data from all context address spaces. There is only one system virtual address space. Another address space, the system physical address space (SPA) defines hardware control registers.

(Col. 18, ll. 29-38)

Thus, the MAPIN and MAPOUT features of *Kaufman* merely allow forked processes to map to a section of context address space of a parent process, and do not provide mapping to the virtual address of a parent process. (*See* Col. 30, ll. 4-14 and Col. 33, ll. 25-30 of *Kaufman*).

In the Advisory Action mailed September 13, 2005, the Examiner contended that the virtual address space stores all data from the context address space and that each segment of the context address space is mapped to the system virtual address space. The Examiner further asserted that when a process's context address space is mapped, the process's virtual address space is also mapped. That assertion is false for at least two reasons. First, *Kaufman* does not at all teach or suggest that the virtual address space is mapped out with the context

address space. *Kaufman* merely discloses that the VM system executes the steps of MAPOUT as a user entry for mapping out a file range from a process's context address space (*See* Col. 34, ll. 10-15 of *Kaufman*). Second, there is no per-process virtual address space in *Kaufman*. *Kaufman* explicitly states that there is only one virtual address space. (*See* Col. 18, ll. 35-36 of *Kaufman*).

>   C. <u>*Hetherington* and *Kaufman* fail to disclose or suggest further limitations of claims 2 and 16</u>

Applicants respectfully submit that the Examiner has omitted one or more essential elements needed for a *prima facie* rejection. The cited art, alone or in combination, fails to teach or suggest every limitation of the claims. For example, the combination of *Hetherington* and *Kaufman* fails to teach or suggest at least "transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data," as recited in claims 2 and 16. The Examiner asserts that the interprocess communication (IPC) facility of *Hetherington* teaches this limitation. (*See* Col. 4, ll. 13-19 of *Hetherington*). However, IPC, including remote procedure call (RPC), typically employs data serialization. (*See* page 2, ll. 22-27 of the patent application). Data is serialized at a first end, transferred between processes, and then deserialized at the other end. *Hetherington* fails to disclose that its IPC facility does not employ data serialization, and therefore teaches away from the limitations of claims 2 and 16.

For at least these reasons, Applicants respectfully submit claims 1-11 and 15-21 are allowable over *Hetherington* in view of *Kaufman*.

II.   <u>*Kaufman* in view of *Heatherington* fails to disclose or suggest claims 12-14</u>

Applicants respectfully submit that the Examiner has omitted one or more essential elements needed for a *prima facie* rejection. The cited art, alone or in combination, fails to teach or suggest every limitation of the claims. For example, the combination of *Kaufman* and *Hetherington* fails to teach or suggest at least mapping a user-specific process to virtual addresses that mirror addresses of a global process, as claimed in claim 12.

Just as *Kaufman* and *Heatherington* fails to teach or suggest "mapping a user-specific process so that it overlays virtual addresses of the master process," the cited art also fails to teach or suggest mapping "a user-specific process to virtual addresses that mirror virtual addresses of the global process."

For at least this reason, Applicants respectfully submit claims 12-14 are allowable over *Kaufman* in view of *Hetherington*.

Appellant respectfully requests that the Board reverse the rejection of claims 1-21.

VIII. CONCLUSION:

For the foregoing reasons, Appellants respectfully submit that the rejections posed by the Examiner are improper as a matter of law and fact. Accordingly, Appellants respectfully request the Board reverse the rejections of claims 1-21.

Respectfully submitted,

  /Christopher P. Rauch/   (Reg. No. 45,034)

Christopher P. Rauch

SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box #061080
Wacker Drive Station - Sears Tower
Chicago, IL 60606-1080
Telephone 312/876-2606
Customer #58328
Attorneys for Appellants

<u>**CLAIMS APPENDIX**</u>

1. (Original) In a computer system, a method for providing for concurrent subprocessing of a master process, the method comprising the steps of:

interfacing with a master process when a user-specific operation is encountered;

mapping a user-specific process so that it overlays virtual addresses of the master process; and

processing the user-specific operation in the user-specific process.

2. (Original) The method of claim 1, further comprising the step of:

transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data.

3. (Original) The method of claim 1, further comprising the step of:

providing an interface for the user-specific process that mirrors an interface for the master process.

4. (Previously presented) The method of claim 1 wherein the master process is a global locale process and the user-specific process is a locale-specific process.

5. (Original) The method of claim 1 wherein the user-specific process is mapped after the user-specific operation is encountered.

6. (Original) The method of claim 1 wherein the user-specific process is mapped before the user-specific operation is encountered.

7. (Original) The method of claim 1 further comprising the step of:

returning processing to the master process after processing the user-specific operation in the user-specific process.

8. (Previously presented) A computer-readable medium comprising computer instructions that facilitate concurrent handling of subprocesses in a system that utilizes a global process, wherein the instructions, when executed, cause the system to perform the steps of:

interfacing with the global process when a user-specific operation is encountered;

mapping a plurality of concurrent user-specific processes, wherein each user-specific process is mapped to virtual addresses that are equivalent to virtual addresses of the global process; and

processing the user-specific operation in one of the user-specific processes.

9. (Previously presented) The computer-readable medium of claim 8, wherein the instructions, when executed, provide each of the plurality of concurrent user-specific processes with an interface that is identical to an interface of the global process.

10. (Previously presented) The computer-readable medium of claim 9, wherein the instructions, when executed, cause the system to perform the step of mapping subprocesses within each of the plurality of user-specific processes, the subprocesses being mapped to virtual addresses that are equivalent to virtual addresses for user-specific operations of the global process.

11. (Previously presented) The computer-readable medium of claim 10, wherein the instructions, when executed, cause the system to perform the step of returning processing to the global process after execution of the subprocesses is complete.

12. (Previously presented) A computer system for enabling concurrent multiple subprocess handling in a global process environment, the system comprising:

a global process; and

a virtual memory separator that maps a user-specific process to virtual addresses that mirror virtual addresses of the global process, the user-specific process having an interface that mirrors an interface of the global process.

13. (Previously presented) The computer system of claim 12 wherein the global process is a global locale process and wherein the user-specific process is a locale-specific process.

14. (Previously presented) The computer system of claim 12 wherein the global process is a global daemon process and wherein the user-specific process is a user-specific daemon process.

15. (Original) An apparatus for conducting multi-user concurrent handling of subprocesses, the apparatus comprising:

means for interfacing with a master process when a user-specific operation is encountered;

means for mapping a user-specific process so that it overlays virtual addresses of the master process; and

means for processing the user-specific operation in the user-specific process.

16. (Original) The apparatus of claim 15, further comprising:

means for transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data.

17. (Original) The apparatus of claim 15, further comprising:

means providing an interface for the user-specific process that mirrors an interface for the master process.

18. (Previously presented) The apparatus of claim 15 wherein the master process is a global locale process and the user-specific process is a locale-specific process.

19. (Original) The apparatus of claim 15 wherein the user-specific process is mapped after the user-specific operation is encountered.

20. (Original) The apparatus of claim 15 wherein the user-specific process is mapped before the user-specific operation is encountered.

21. (Original) The apparatus of claim 15, further comprising:

means for returning processing to the master process after the user-specific operation is executed in the user-specific process.

EVIDENCE APPENDIX

Appellants do not submit additional evidence with this appeal brief and no additional evidence has been submitted during prosecution.

RELATED PROCEEDINGS APPENDIX

Appellants are not aware of any related appeals or interferences with regard to the present application.